2023012351 刘奕彤

第一章

1. 操作系统是管理计算机硬件和软件资源的系统软件，充当用户与计算机硬件之间的接口。

4. 实时操作系统是一种保证任务在严格时间限制内完成的操作系统。适用于需要即时响应的场景。

5. 互斥访问是并发环境下保证资源独占访问的机制，防止多个进程/线程同时访问共享资源导致数据不一致

二.

2. A (2)
   B (2)
   C (4)

3. A (2)
   B (3)
   C (4)

三.

1. 硬件. 软件.
3. 硬件资源. 软件资源
5. 共享性. 异步性.

第二章

一.

3. 临界区. 是进程中访问共享资源的代码段，必须保证同一时刻仅有一个进程/线程执行该段代码，以防止数据不一致或竞态条件.

4. 进程同步. 是协调多个并发进程的执行顺序，确保它们按照特定规则有序访问共享资源或协作完成任务.

二. 1. ~~(1)~~ (3) (4) (2)

(3) → (4)

(4) → (2)

3. A (2)
   B (2)
   C (4)
   D (3)

三. 1. 动态性. 异步性. 结构化性

2. 动态. 静态.

3. 6. 间接制约

8. 共享存储. 消息传递. 管道通信

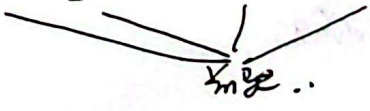
9. 运行. 就绪. 阻塞.

五.

2. S_1 是 S_2 和 S_3 的前驱. ($S_1 \to S_2$, $S_1 \to S_3$)

S_2 与 S_3 的关系

S_4 无. $S_2 \to S_4$, $S_2 \to S_5$, $S_3 \to S_6$. $S_4, S_5, S_6 \to S_7$



```
Semaphore a=0;
Semaphore b=0;
Semaphore c=0;
Semaphore d=0; Semaphore g=0;
Semaphore e=0; Semaphore h=0;
Semaphore f=0;
```

```
process_S1(){
    V(a);
    V(b);
}
```

```
process_S2(){
    P(a);
    V(c);
    V(h);
}
```

```
process_S3(){
    P(b);
    V(d);
}
```

```
process_S4(){
    P(c);
    V(e);
}
```

```
process_S5(){
    P(h);
    V(f);
}
```

```
process_S6(){
    P(h);
    V(f);
}
```

```
process_S7(){
    P(d);
    V(g);
}
```

```
process_S8(){
    P(e);
    P(f);
    P(g);
}
```

3. semaphore mutex = 1;
   semaphore empty = 5;
   semaphore apple = 0;
   semaphore orange = 0;

```
void father(){
    while(true){
        准备水果类型(苹果或桔子);
        P(empty);
        P(mutex);
        放入水果.
        if(水果是苹果) V(apple);
        else V(orange);
        V(mutex);
    }
}
```

```
void son(){
    while(true){
        P(orange);
        P(mutex);
        取桔子;
        V(empty);
        V(mutex);
        吃桔子;
    }
}
```

```
void daughter(){
    while(true){
        P(apple);
        P(mutex);
        取苹果;
        V(empty);
        V(mutex);
        吃苹果;
    }
}
```

4.

```
Semaphore empty1 = 1
Semaphore full1 = 0
semaphore empty2 = 1
semaphore full2 = 0

void PA() {
    while(有记录未读){
        从磁盘读取一个记录;
        P(empty1);
        将记录存入缓冲区1;
        V(full1);
    }
}

void PB(){
    while(需要处理){
        P(full1);
        P(empty2);
        从缓冲区1复制到缓冲区2;
        V(empty1);
        V(full2);
    }
}

void Pc(){
    while(需要打印){
        P(full2);
        从缓冲区2打印记录;
        V(empty2);
    }
}
```

5.

```
Semaphore customers = 0;
semaphore barber = 0;
semaphore mutex = 1;
semaphore payment = 0;
semaphore sofa = N;
int waiting = 0;

void barber(){
    while(true){
        P(customers);
        P(mutex);
        waiting--;
        V(sofa);
        V(barber);
        V(mutex);
        理发;
        P(payment);
        接受付款;
    }
}

void customer(){
    P(mutex);
    if(waiting >= N){
        V(mutex);
        离开;
        return;
    }
    waiting++;
    V(customers);
    P(sofa);
    V(mutex);
    P(barber);
    离开沙发;
    坐到理发椅;
    接受理发服务;
    V(payment);
    付款后离开;
}
```

第三章 2023012551 刘奕彤

一、2. 处理机调度是操作系统为多个进程分配CPU使用权的过程，核心目标是提高CPU利用率、优化系统吞吐量、并确保公平性.

3. 周转时间指作业从提交到完成所经历的总时间，用于衡量系统对作业的响应效率.

4. 死锁是多个进程因竞争资源而无限等待的状态，每个进程持有其他进程所需的资源且不释放.

二、

1. (2) (5) (3) (6) (4) (2)

3. (5) (2) (1) (3) (2)

4. (2)(1)

三、

2. 提交、后备、完成

6. 预防死锁、避免死锁、检测与解除死锁、忽略死锁.

五、

1. ① 先来先服务(FCFS)

A → B → C → D → E.

进程	完成时间	周转时间	带权周转时间
A	3	3	1.0
B	9	7	7/6 ≈ 1.17
C	13	9	$\frac{9}{4}$ = 2.25
D	18	12	$\frac{12}{5}$ = 2.4
E	20	12	$\frac{12}{2}$ = 6.0

平均周转时间: $\frac{3+7+9+12+12}{5} = \frac{43}{5} = 8.6$

平均带权……: $\frac{1.0+1.17+2.25+2.4+6.0}{5} = 2.564$

② SJF (非抢占)

A → B → E → C → D

与①中对应.

A	3	3	1.0
B	9	7	1.17
C	15	11	2.75
D	20	14	2.8
E	11	3	1.5

平均周转: $\frac{3+7+11+14+3}{5} = \frac{38}{5} = 7.6$

平均带权~: $\frac{1.0+1.17+2.75+2.8+1.5}{5} = 1.844$

③ SJF (抢占)

A → B → C → E → C → D

A	3	3	1.0
B	9	7	1.17
C	12	8	2.0
D	17	11	2.2
E	10	2	1.0

平均周转~: $\frac{3+7+8+11+2}{5} = \frac{31}{5} = 6.2$

平均带权~: 7.37 [illegible]

7 + 4.65 + 1.17 = 12.82 ; 12.82 ÷ 5 = 2.564

1.0 + 1.17 = 2.17 ; + 2.75 = 4.92 ; + 4.3 = 9.22 ; 9.22 ÷ 5 = 1.844

6.2 1.17

7.37 ÷ 5 = 1.474

④ HRRN

A→B→C→E→D

A	3	3	1.0
B	9	7	1.17
C	13	9	2.25
D	20	14	2.8
E	15	7	3.5

平均周转: $\frac{42}{5}=8$

平均带权: $\frac{10.72}{5}=2.144$

$\begin{array}{r} 42 \\ +0.3 \\ \hline 10.72 \end{array}$

$\begin{array}{r} 2.144 \\ 5\overline{)10.72} \\ \underline{10} \\ 72 \\ \underline{70} \\ 22 \end{array}$

⑤ (RR,时间片1)

A	3	3	1.0
B	14	12	2.0
C	16	12	3.0
D	20	14	2.8
E	11	3	1.5

平均周转: $\frac{44}{5}=8.8$

平均带权: $\frac{10.3}{5}=2.06$

2.

⑥ ①0~10ms A: 20-0-10=10
B: 50-0-10=40
C: 50-0-15=35
调度A

②10~20ms B: 50-10-10=30
C: 50-10-15=25
无作业需调行

③20~30ms: A: 40-20-10=10
B: 50-20-10=20
C: 50-20-15=15
调度A

④30~40ms B: 50-30-10=10
C: 50-30-15=5
调度C

⑤40~50ms B: ~~50ms~~ 50-40-10=0 调度B.

∴ 0~10: A 20~30: A 30~40: C 40~50: B

2023012557 刘奥彤

3.(1) 进程

P_0 [0-0, 0-0, 4-1, 4-2] = [0, 0, 3, 2]

P_1 [2-1, 7-7, 5-5, 0-0] = [1, 0, 0, 0]

P_2 [3-2, 6-3, 10-5, 10-6] = [1, 3, 5, 4]

P_3 [0-0, 9-6, 8-5, 4-2] = [0, 3, 3, 2]

P_4 [0-0, 6-6, 6-5, 10-6] = [0, 0, 1, 4]

P_0: Need = [0, 0, 1, 2] ≤ Available = [1, 6, 2, 2]

Available = [1+0, 6+0, 2+3, 2+2] = [1, 6, 5, 4]

P_3: [0, 6, 3, 2] ≤ [1, 6, 5, 4]

A... = [1, 9, 8, 6]

P_4: [0, 6, 5, 6] ≤ [1, 9, 8, 6]

A.... = [1, 9, 9, 10]

P_1: [1, 7, 5, 0] ≤ [1, 9, 9, 10]

A... = [1+1, 9+0, 9+0, 10+0] = [2, 9, 9, 10]

P_2: [2, 3, 5, 6] ≤ [2, 9, 9, 10]

∴安全序列为 $P_0 \rightarrow P_3 \rightarrow P_4 \rightarrow P_1 \rightarrow P_2$.

∴该状态安全.

(2) Request ≤ Need? [1, 2, 2, 2] ≤ [2, 3, 5, 6] → 是

Request ≤ Available? [1, 2, 2, 2] ≤ [1, 6, 2, 2] → 是

假设:

Available = [1-1, 6-2, 2-2, 2-2] = [0, 4, 0, 0]

Need = [2-1, 3-2, 5-2, 6-2] = [1, 1, 3, 4]

Allocation = [1+1, 3+2, 5+2, 4+2] = [2, 5, 7, 6]

可用资源 [0, 4, 0, 0] 无法满足任何进程 Need.

∴无法找到安全序列. 拒绝 P_2 请求.

(3) Available = [0, 4, 0, 0] 所有进程 Need 均无法满足.

P_0: [0, 0, 1, 2] > [0, 4, 0, 0] (第3, 4资源不同).

同理. 其他进程也无法满足

∴系统立即进入死锁状态